Національний університет “Одеська політехніка”
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Алгоритмізація та програмування»

Тема «Програмування динамічної структури даних – однозв’язний список»

Студентки 1 курсу AI-212 групи
Спеціальності 122 – «Комп’ютерні науки»
Козуб К.О
Керівник ст.викл., к.т.н. Манікаєва О. С.

Національна шкала _____
Кількість балів: _____
Оцінка: ECTS _____

Члени комісії

_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали)
_____ _____
(підпис) (прізвище та ініціали

м. Одеса – 2022 рік

Одеський національний політехнічний університет
Інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студенткі Козуб Катерини Олексіївні група АІ-212

1. Тема роботи
«Програмування динамічної структури даних – однозв’язний список»

2. Термін здачі студентом закінченої роботи 03.06.2021

3. Початкові дані до проекту (роботи)
Варіант 10
Структура: вид спорту, прізвище спортсмена, яке місце зайняв, країна, результат, особистий рекорд;
Програма повинна виконувати: додавання елемента; видалення елемента; можливість коригування даних; виведення всіх даних; початкове формування даних про всі спортсменів у вигляді таблиці; виведення даних про введені види спорту; виведення даних про спортсменів, які поліпшили свій особистий рекорд; пошук всіх спортсменів з введеної країни; виведення зведеної таблиці із зазначенням країни, кількості перших, других, третіх місць; сортування по полю «вид спорту
4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)
Вступ. Теоретичні відомості про однозв’язний список. Програмна реалізація– однозв’язний список. Інструкція користувача. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Блок-схема алгоритму – 1 аркуш формату А1.

Завдання видано 16.03.21 _____
(підпис викладача)

Завдання прийнято до виконання 16.03.21 _____
(підпис студента)

АНОТАЦІЯ

Розглянуто безліч шляхів вирішення проблем, які виникають при розробці динамічних структур за допомогою процедурного підходу.

Реалізація динамічної структури даних розглянута більш детально також пропонується деяке змішування С і С ++ для більш гнучкого виконання програми.

А однозв’язний список був розроблений як шаблон для майбутнього виконання гнучкості.

ЗМІСТ

Вступ

Мета роботи: метою курсової роботи є закріплення і поглиблення знань, одержаних студентами в курсі «Алгоритмізація та програмування», розвиток навичок при виборі представлення початкових даних, вдосконалення техніки використання засобів тестування і налагоджування програми, оформлення документації на програмну розробку.

У сучасній розробці досить часто доводиться працювати зі структурами, розмір яких може бути відомий раніше часу. Тому є динамічні структури даних які надзвичайно гнучко впораються з цією проблемою**.**
Існують такі види ДСД:
-Черга(з пріоритетом,кільцева, двобічна)
-Дерево
-Стек
-Список(однозв’язний, циклічний,двозв’язний)

Існує декілька особливостей розробки динамічних структур даних за допомогою процедурного підходу. Першу чергу розробка процедурним підходом зменшує єдність функцій як до визначеною структури даних також це необхідність постійно передавати аргументом фіксовану структуру. Також процедурний підхід у розробці динамічних структур потребує тісної взаємодії з посиланнями та raw-pointers, що може призвести до проблем(якщо розробник ,наприклад, забуде висвободити пам’ять назад до кучі(heap)).

Основні етапи розробки програмного продукту за допомогою процедурного підходу виглядають приблизно так:
-Обробка вхідних даних(замовник, тощо)

-Розробка алгоритму програми

-Програмування, скриптування

-Відладка

-Тестування

-Документація

Деякі відмінності могуть з’явитися на рівні побудови архітектури, проте вони не є критично відмінними від побудови іншого програмного продукту. Функціональне програмування використовується для низькорівневого програмування де використання об’єктів є надмірним та ресурсоємким.

Теоротичні відомості про однозв’язний список

Однозв’язний список – дані динамічної структури,що являють собою сукупність лінійно зв’язаних однорідних елементів, до якихдозволяється додавати елементи на голови (початку) або в кінець (хвіст) списку,між будь-якими двома іншими і видаляти будь-який елемент (рис.1.).

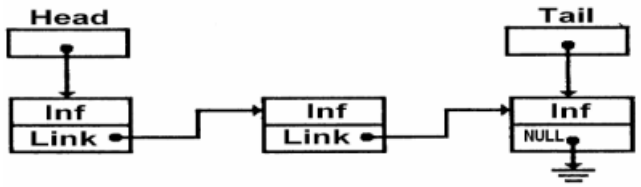


Рис.1. Схематичне зображення однозв'язного лінійного списку

Для роботи з однозв’язним списком потрібні такі вказівники:

1) на голову списку Head;

2) кінець списку Tail, але можна обійтися й без нього;

3) k-й елемент списку;

4) тимчасовий для виділення пам’яті під елементи, які додаються, і для звільнення елементів, що видаляються (ідентифікатор p).

Особливості кодування. Базові операції над списком

Однозв’язний список може зберігатися:

-в оперативній пам’яті;

-у файлі.

В однозв’язному списку виділяють окремий елемент (Element), який ще називається вузол (Node).

Кожен елемент (вузол) однозв’язного списку складається з двох частин:

-дані. Це може бути змінна будь-якого примітивного типу (int, double, char, …), об’єкт класу чи складна структура. Дані можуть складатися з декількох полів (змінних);

-адреса або позиція наступного елементу. Якщо однозв’язний список зберігається в оперативній пам’яті, то це є покажчик (адреса) на такий самий елемент (вузол). Якщо однозв’язний список зберігається в файлі, то це є позиція наступного елементу в файлі.

Реалізувати вузол можна з допомогою класу або структури. Також можна створювати однозв’язний список на основі шаблонного класу, який оперує деяким узагальненим типом T.

Сукупність елементів (вузлів), в яких покажчик попереднього елементу вказує на наступний елемент, утворює однозв’язний список.

Для списку можна виділити наступні основні операції:

- формування списку;
- додавання елементу в кінець списку;
- вставка елементу в задану позицію списку;
- видалення елементу зі списку з заданої позиції;
- очищення списку;

- заміна елементу в списку;
- пошук елементу в списку за індексом чи деяким критерієм.

Додавання елемента до кінця списку

При додаванні елементу в кінець порожнього списку виділяють наступні операції (рисунок 4):

2. Створити новий елемент. Тут потрібно виділити пам’ять для нового елементу та заповнити її деякими даними (_data). Для цього оголошується покажчик (наприклад, elem):

```
Element<T>* elem = new Element<T>;
elem->data = _data;
```

2. Встановити покажчик next в нульове значення

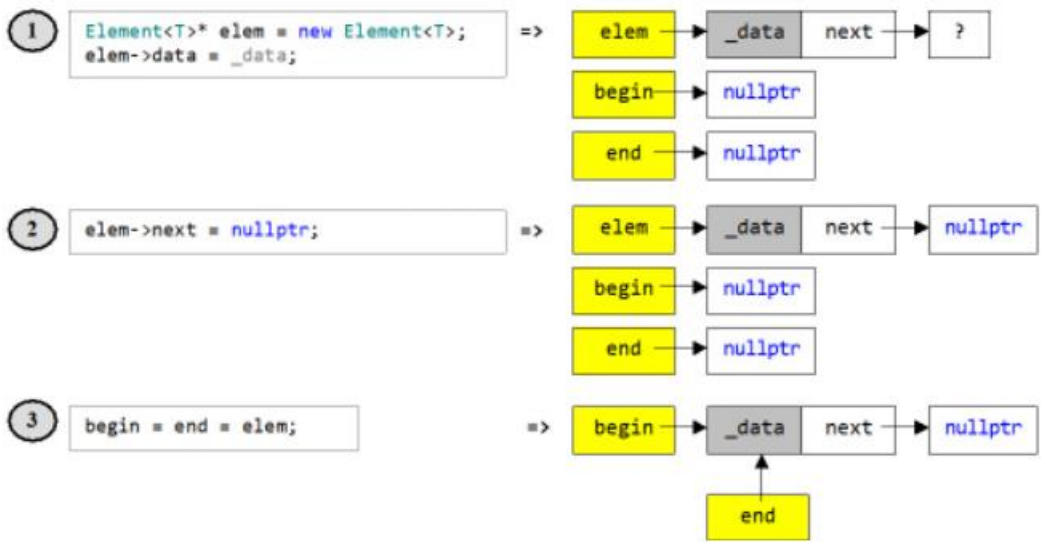
```
elem->next = nullptr;
```

3. Встановити покажчики begin та end рівними значенню покажчика elem.

```
begin = end = elem;
```

4. Збільшити кількість елементів у списку на 1

```
count++;
```



Додавання елемента до кінця існуючого списку

Якщо список вже існує, то послідовність кроків що додають елемент до списку, наступна (рисунок 5):

2. Створити новий елемент та заповнити його даними (рисунок 5). Заповнити поля data та next. Створення здійснюється таким самим чином як у випадку з порожнім списком:

```
Element<T>* elem = new Element<T>;
elem->data = _data;
elem->next = nullptr;
```

2. Встановити покажчик next елементу, на який вказує покажчик end, в значення адреси елементу elem (рисунок 6)

```
end->next = elem;
```

3. Встановити покажчик end рівним значенню покажчика elem

```
end = elem;
```

4. Збільшити кількість елементів у списку на 1

```
count++;
```

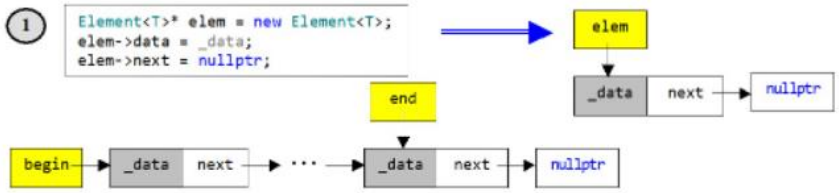


Рисунок 5. Створення елементу та заповнення його даними

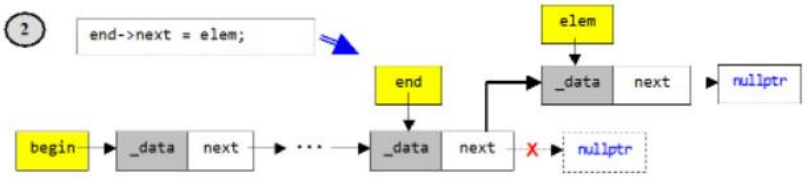
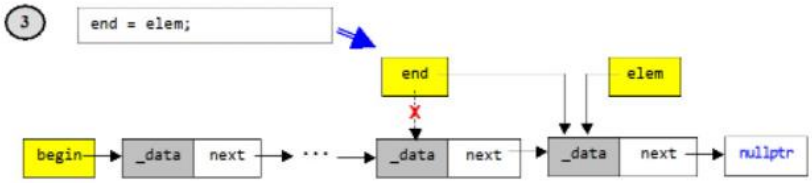


Рисунок 6. Зміна покажчика next елементу end



Видалення елемента зі списку у заданій позиції

Видалення елемента зі списку у заданій позиції

При вставці елементу в задану позицію списку розглядаються 2 варіанти:

- список порожній (немає елементів);
- список має хоча б один елемент.

Якщо список порожній, то вставка замінюється додаванням першого елементу.

Якщо у списку є елементи, то тут розглядаються 3 варіанти:

- вставка елементу в першу позицію списку;
- вставка елементу в середині списку;
- вставка елементу за останнім елементом списку. Якщо відбувається вставка за останнім елементом списку, то це є додавання елементу в кінець списку

Програмна реалізація однозв’язний список

2.1 Меню

```
void printMenu()
{
	system(«cls»);
	printf(«%62sСписок доступних варіантів\n», «»);
	printSeparator(‘~’);
	printf(«%20s 1) Вивести список на екран	2) Додати новий запис	3) Видалити існуючий запис\n», «»);
	printf(«%20s 4) Коригувати дані	5) Дані про всіх спортсменів	6) Дані за видом спорту\n», «»);
	printf(«%20s 7) Спортсмени з новим рекордом	8) Пошук за країною	9) Рейтинг країн\n», «»);
	printf(«%20s10) Сортування за назвою спорту	11) Зберегти та вийти	12) Вийти без збереження\n», «»);
	printSeparator(‘~’);
}
```

2.2 Додавання елемента

```
void addNode(ListHead& listHead, ListNode* node, int position)
{
	listHead.elementCount++;
	node->next = nullptr;
	if (listHead.first == nullptr)
	{
		listHead.first = node;
	}
	else
	{
		ListNode* current = listHead.first;
		ListNode* previous = nullptr;
		for (int i = 1; i < position; i++)
		{
			previous = current;
			current = current->next;
		}
		node->next = current;
		(previous == nullptr) ? (listHead.first = node) : (previous->next = node);
	}
}
```

2.3 Видалення елемента

```
void deleteNode(ListHead& listHead, int number)
{
	ListNode* current = listHead.first;
	ListNode* previous = nullptr;
	listHead.elementCount--;
	for (int i = 1; i < number; i++)
	{
		previous = current;
		current = current->next;
```

```
        }
        (previous == nullptr) ? (listHead.first = current->next) : (previous->next = current->next);
        delete current;
}
```

2.4 Виведення всіх даних

```
void printData(ListHead& listHead)
{

        if (listHead.first == nullptr)
        {
                printf(«Список пустий!»);
        }
        else
        {
                printSeparator('-');
                printf(«  Назва спортивної дисципліни  | №  |        Ім'я спортсмена        |»);
                printf(«            Країна            |      Результат      | Особистий рекорд  |\n»);
                ListNode* current = listHead.first;
                printSeparator('-');
                while (current != nullptr)
                {
                        printNode(current);
                        current = current->next;
                }
        }
}
```

2.5 Початкове формування даних про всі спортсменів

```
//Получение информации о всех спортсменах
void sportsmenData(ListHead& listHead)
{
        char** sportsmen = new char* [listHead.elementCount];
        for (int i = 0; i < listHead.elementCount; i++)
        {
                sportsmen[i] = new char[STR_LENGTH];
        }
        int k = 0;
        ListNode* current = listHead.first;
        while (current != nullptr)
        {
                bool isHere = false;
                for (int i = 0; i < k; i++)
                {
                        if (strcmp(current->firstPlaceName, sportsmen[i]) == 0)
                        {
                                isHere = true;
                                break;
                        }
                }
                if (!isHere) sportsmen[k++] = current->firstPlaceName;
                isHere = false;
                for (int i = 0; i < k; i++)
```

```
		{
			if (strcmp(current->secondPlaceName, sportsmen[i]) == 0)
			{
				isHere = true;
				break;
			}
		}
		if (!isHere) sportsmen[k++] = current->secondPlaceName;
		isHere = false;
		for (int i = 0; i < k; i++)
		{
			if (strcmp(current->thirdPlaceName, sportsmen[i]) == 0)
			{
				isHere = true;
				break;
			}
		}
		if (!isHere) sportsmen[k++] = current->thirdPlaceName;
		current = current->next;
	}
	for (int i = 0; i < k; i++)
	{
		printSeparator('-');
		printf(« %-30s», sportsmen[i]);
		bool isFirst = true;
		current = listHead.first;
		while (current != nullptr)
		{
			if (strcmp(current->firstPlaceName, sportsmen[i]) == 0) {
				if (!isFirst) printf(« %30s», «»);
				printf(«  |%-30s |», current->sportName);
				printf(« 1 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n»,
current->firstPlaceResult, current->firstPlaceBest);
				isFirst = false;
			}
			if (strcmp(current->secondPlaceName, sportsmen[i]) == 0) {
				if (!isFirst) printf(« %30s», «»);
				printf(«  |%-30s |», current->sportName);
				printf(« 2 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n»,
current->secondPlaceResult, current->secondPlaceBest);
				isFirst = false;
			}
			if (strcmp(current->thirdPlaceName, sportsmen[i]) == 0) {
				if (!isFirst) printf(« %30s», «»);
				printf(«  |%-30s |», current->sportName);
				printf(« 3 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n»,
current->thirdPlaceResult, current->thirdPlaceBest);
				isFirst = false;
			}
			current = current->next;
		}
```

```
        }
        printSeparator('-');
}
```

2.6 Виведення даних про введені види спорту

```
void sportSearch(ListHead& listHead, char* sportName)
{
        ListNode* current = listHead.first;
        int isPrinted = false;
        while (current != nullptr) {
                if (strcmp(current->sportName, sportName) == 0)
                {
                        if (!isPrinted) printSeparator('-');
                        printNode(current);
                        isPrinted = true;
                }
                current = current->next;
        }
        if (!isPrinted)
                cout << «Нічого не знайдено!» << endl << endl;
}
```

2.7 Пошук всіх спортсменів з введеної країни

```
void getCountryResults(ListHead& listHead)
{
        struct CountryResults
        {
                char country[STR_LENGTH];
                int firstCount = 0;
                int secondCount = 0;
                int thirdCount = 0;
        } *countryResult = new CountryResults[listHead.elementCount * 3];

        ListNode* current = listHead.first;
        int k = 0;
        while (current != nullptr)
        {
                bool isHere = false;
                for (int i = 0; i < k; i++)
                {
                        if (strcmp(current->firstPlaceCountry, countryResult[i].country) == 0)
                        {
                                countryResult[i].firstCount++;
                                isHere = true;
                                break;
                        }
                }
                if (!isHere)
                {
                        strcpy(countryResult[k].country, current->firstPlaceCountry);
                        countryResult[k].firstCount = 1;
                        k++;
```

```
            }
            isHere = false;
            for (int i = 0; i < k; i++)
            {

                if (strcmp(current->secondPlaceCountry, countryResult[i].country) == 0)
                {
                    countryResult[i].secondCount++;
                    isHere = true;
                    break;
                }
            }
            if (!isHere)
            {
                strcpy(countryResult[k].country, current->secondPlaceCountry);
                countryResult[k].secondCount = 1;
                k++;
            }
            isHere = false;
            for (int i = 0; i < k; i++)
            {
                if (strcmp(current->thirdPlaceCountry, countryResult[i].country) == 0)
                {
                    countryResult[i].thirdCount++;
                    isHere = true;
                    break;
                }
            }
            if (!isHere)
            {
                strcpy(countryResult[k].country, current->thirdPlaceCountry);
                countryResult[k].thirdCount = 1;
                k++;
            }
            current = current->next;
        }
        if (k == 0) cout << "Нічого не знайдено!" << endl << endl;
        else printSeparator('-');
        for (int i = 0; i < k; i++)
        {
            printf(" %-30s | Золото: %3d | Срібло: %3d | Бронза: %3d |\n",
                countryResult[i].country,
                countryResult[i].firstCount,
                countryResult[i].secondCount,
                countryResult[i].thirdCount);
            printSeparator('-');
        }


}
```

ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 Меню

```
Список доступних варіантів

 1) Вивести список на екран        2) Додати новий запис          3) Видалити існуючий запис
 4) Коригувати дані                5) Дані про всіх спортсменів   6) Дані за видом спорту
 7) Спортсмени з новим рекордом    8) Пошук за країною            9) Рейтинг країн
10) Сортування за назвою спорту   11) Зберегти та вийти          12) Вийти без збереження
```

3.2 Вивести список на екран

Назва спортивної дисципліни	№	Ім'я спортсмена	Країна	Результат	Особистий рекорд
Біг на 100 метрів	№ 1	Олійник Василь	Україна	Результат: 10.40	Рекорд: 10.50
	№ 2	Бернар Ріє	Франція	Результат: 10.80	Рекорд: 10.30
	№ 3	Міюкі Масаюкі	Японія	Результат: 11.00	Рекорд: 11.00
Біг на 200 метрів	№ 1	Дзьнь Мінь	Китай	Результат: 18.00	Рекорд: 17.00
	№ 2	Бернар Ріє	Франція	Результат: 19.00	Рекорд: 19.00
	№ 3	Олійник Василь	Україна	Результат: 20.65	Рекорд: 20.66
Сортивна Хода	№ 1	Ягамі Рінтаро	Японія	Результат: 38.00	Рекорд: 37.00
	№ 2	Бернар Ріє	Франція	Результат: 40.00	Рекорд: 39.00
	№ 3	Віктор Храпко	Україна	Результат: 41.00	Рекорд: 35.00
Біг на 400 метрів	№ 1	Хуліо Гонзалес	Мексика	Результат: 37.50	Рекорд: 37.10
	№ 2	Бернар Ріє	Франція	Результат: 38.10	Рекорд: 38.50
	№ 3	Віктор Храпко	Україна	Результат: 38.50	Рекорд: 39.00

Натисніть будь-яку клавішу, щоб продовжити...

3.3 Додавання новий запис

Введіть назву виду спорту:	Біг на 100 метрів
Введіть ім'я спортсмена, що посів перше місце:	Олійник Василь
Введіть країну спортсмена, що посів перше місце:	Укрвїна
Введіть результат спортсмена, що посів перше місце:	10.40
Введіть персональний рекорд спортсмена, що посів перше місце:	10.50
Введіть ім'я спортсмена, що посів друга місце:	Бернар Ріє
Введіть країну спортсмена, що посів друга місце:	Франція
Введіть результат спортсмена, що посів друга місце:	10.80
Введіть персональний рекорд спортсмена, що посів друга місце:	10.30
Введіть ім'я спортсмена, що посів трете місце:	Міюкі Масаюкі
Введіть країну спортсмена, що посів трете місце:	Японія
Введіть результат спортсмена, що посів трете місце:	11.00
Введіть персональний рекорд спортсмена, що посів трете місце:	11.00

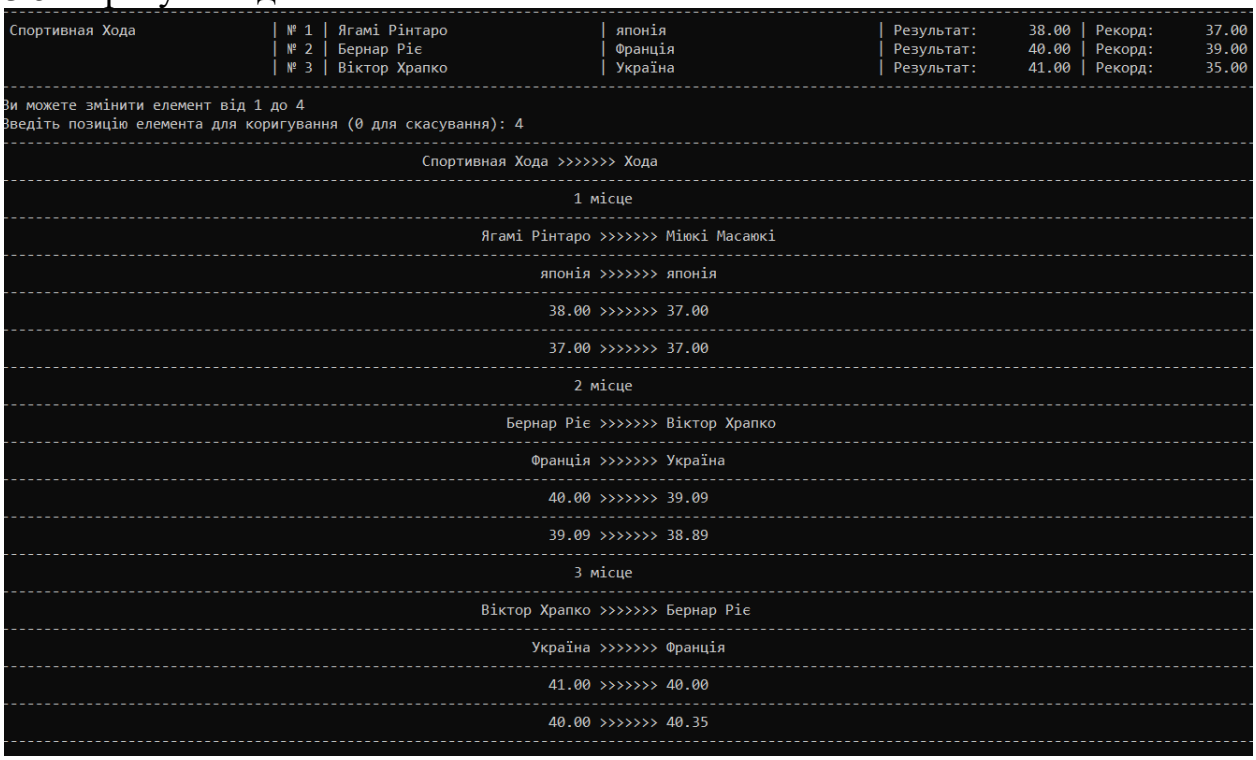
Натисніть будь-яку клавішу, щоб продовжити...

3.4 Видалити існуючий запис

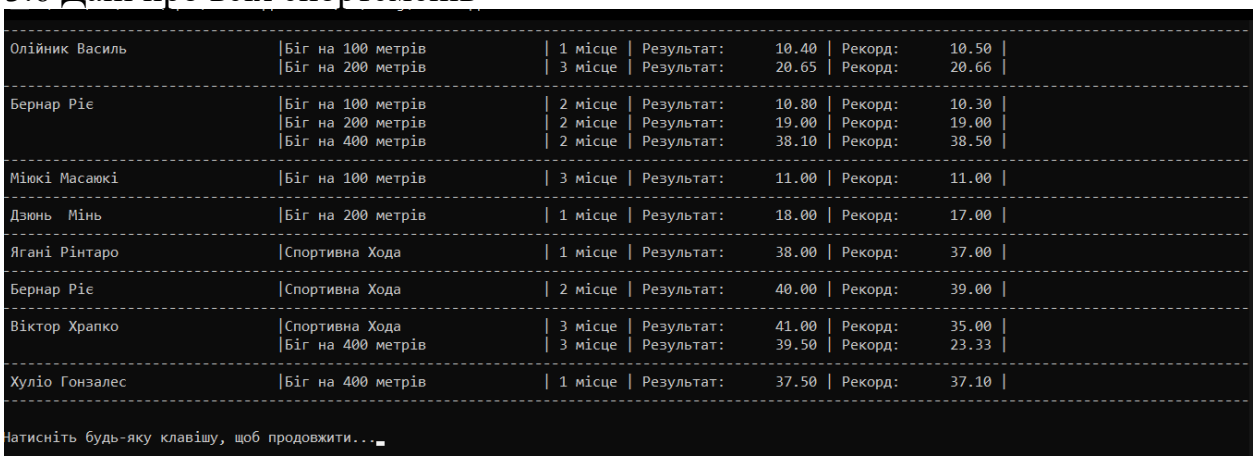
Назва спортивної дисципліни	№	Ім'я спортсмена	Країна	Результат	Особистий рекорд
Спорт 1	№ 1	Спортсмен 1	Країна 1	Результат: 9.00	Рекорд: 10.00
	№ 2	Спортсмен 2	Країна 2	Результат: 11.00	Рекорд: 11.00
	№ 3	Спортсмен3	Країна 3	Результат: 12.00	Рекорд: 12.00
Біг на 100 метрів	№ 1	Олійник Василь	Україна	Результат: 10.40	Рекорд: 10.50
	№ 2	Бернар Ріє	Франція	Результат: 10.80	Рекорд: 10.30
	№ 3	Міюкі Масаюкі	японія	Результат: 11.00	Рекорд: 11.00
Біг на 200 метрів	№ 1	Дзюнь Мінь	Китай	Результат: 18.00	Рекорд: 17.00
	№ 2	Брнар Ріє	Франція	Результат: 19.00	Рекорд: 19.00
	№ 3	Олійник Василь	Україна	Результат: 20.65	Рекорд: 20.66
Біг на 400 метрів	№ 1	Хуліо Гонзалес	Мексика	Результат: 37.50	Рекорд: 37.10
	№ 2	Бернар Ріє	Франція	Результат: 38.10	Рекорд: 38.50
	№ 3	Віктор Храпко	Україна	Результат: 38.50	Рекорд: 39.00
Спортивная Хода	№ 1	Ягамі Рінтаро	японія	Результат: 38.00	Рекорд: 37.00
	№ 2	Бернар Ріє	Франція	Результат: 40.00	Рекорд: 39.00
	№ 3	Віктор Храпко	Україна	Результат: 41.00	Рекорд: 35.00

Ви можете видалити елемент від 1 до 5
Введіть позицію елемента для видалення (0 для скасування): 1

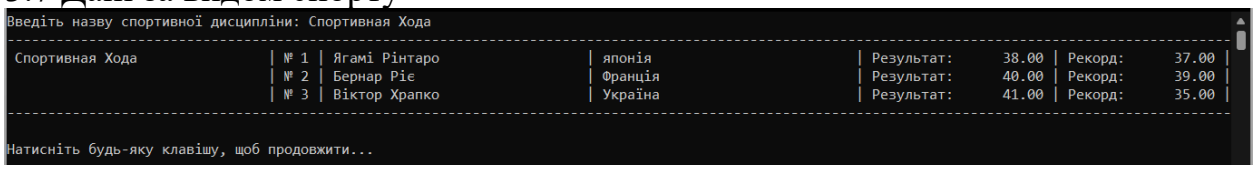
3.5 Коригувати дані



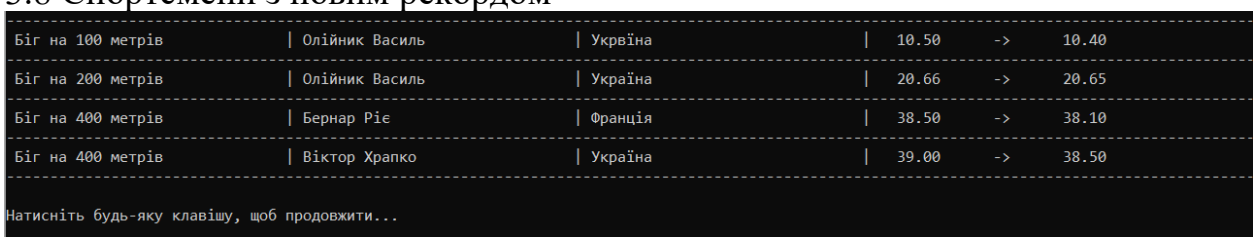
3.6 Дані про всіх спортсменів



3.7 Дані за видом спорту



3.8 Спортсмени з новим рекордом



3.9 Пошук за країною

```
Введіть назву країни: Україна
------------------------------------------------------------------
Олійник Василь
------------------------------------------------------------------
Віктор Храпко
------------------------------------------------------------------

Натисніть будь-яку клавішу, щоб продовжити...
```

3.10 Рейтинг країн

```
-------------------------------------------------------------------------------------------------
 Україна                        | Золото:   1 | Срібло:   0 | Бронза:   2 |
-------------------------------------------------------------------------------------------------
 Франція                        | Золото:   0 | Срібло:   4 | Бронза:   0 |
-------------------------------------------------------------------------------------------------
 японія                         | Золото:   1 | Срібло:   0 | Бронза:   1 |
-------------------------------------------------------------------------------------------------
 Китай                          | Золото:   1 | Срібло:   0 | Бронза:   0 |
-------------------------------------------------------------------------------------------------
 Мексика                        | Золото:   1 | Срібло:   0 | Бронза:   0 |
-------------------------------------------------------------------------------------------------
 Україна                        | Золото:   0 | Срібло:   0 | Бронза:   1 |
-------------------------------------------------------------------------------------------------

Натисніть будь-яку клавішу, щоб продовжити...
```

Висновок

Під час виконання курсової роботи було реалізовано однозв’язний спискок та його функціонал для імітації взаємодії з базою даних ДАІ, було направлено сили на створення, у якійсь мірі, найбільш «дружелюбного» інтерфейсу взаємодії між користувачем та програмою. В ході виконання курсової роботи було дуже поглиблено знання у роботі з бінарними файлами.

Перелік використаних джерел

1. https://www.wiki.uk-ua.nina.az/%D0%9E%D0%B4%D0%BD%D0%BE%D0%B1%D1%96%D1%87%D0%BD%D0%BE_%D0%B7%D0%B2'%D1%8F%D0%B7%D0%B0%D0%BD%D0%B8%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA.html
2. https://www.bestprog.net/uk/2022/02/11/c-linear-singly-linked-list-general-information-ua/
3. Документація Microsoft Visual Studio C++ Microsoft C/C++ Documentation | Microsoft Docs

ДОДАТОК А

КОД ПРОГРАМИ

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <ostream>

#define STR_LENGTH 30

using namespace std;


//Структура для хранения единичногоузла списка
typedef struct ListNodeType
{
	char sportName[STR_LENGTH];
	char firstPlaceName[STR_LENGTH];
	char firstPlaceCountry[STR_LENGTH];
	double firstPlaceResult;
	double firstPlaceBest;
	char secondPlaceName[STR_LENGTH];
	char secondPlaceCountry[STR_LENGTH];
	double secondPlaceResult;
	double secondPlaceBest;
	char thirdPlaceName[STR_LENGTH];
	char thirdPlaceCountry[STR_LENGTH];
	double thirdPlaceResult;
	double thirdPlaceBest;
	ListNodeType* next;
} ListNode;

//Структура хранящая информацию о начальном элементе и их колличестве
typedef struct ListHeadType
{
	ListNode* first = nullptr;
	int elementCount = 0;
} ListHead;


//Ожидание нажатия любой клавиши
void pause();

//Ручной ввод данных
ListNode* getNode();

//Функция безопасного ввода дробного числа
void getDouble(const char*, double&);
```

```
//Функция безопасного ввода целого числа
void getInteger(const char*, int&);

//Функция безопасного строки
void getString(const char*, char*);

//Вывести на экран данные одного узла списка
void printNode(ListNode*);

//Добавить элемент в список
void addNode(ListHead&, ListNode*, int);

//Вывести данные всего списка
void printData(ListHead&);

//Вывести главное меню
void printMenu();

//Удалить элемент списка по индексу
void deleteNode(ListHead& listHead, int number);

//Сохранить список в бинарный файл
void saveToFile(ListHead& listHead);

//Считать данные с бинарного файла
void readFromFile(ListHead& listHead);

//Получение информации о всех спортсменах
void sportsmenData(ListHead& listHead);

//Сортировать сортивные дисциплины по названию
void sortByName(ListHead& listHead);

//Напечатать разделитель
void printSeparator(const char symb);

//Редактировать данные элемента по индексу
void editNode(ListHead& listHead, int number);

//Поиск спортивной дисциплины по названию
void sportSearch(ListHead& listHead, char* sportName);

//Поиск спотсменов по стране
void countrySearch(ListHead& listHead, char* countryName);

//Поиск спортсменов, улучшевших свой результат
void newBest(ListHead& listHead);

//Получить информацию об общих результатах страны
void getCountryResults(ListHead& listHead);

int main()
```

```
{
	system("chcp 1251");
	system("cls");
	system("mode con cols=150");
	ListHead competitionList;
	readFromFile(competitionList);
	int action;
	int number;
	char tmp[STR_LENGTH];
	do
	{
		printMenu();
		getInteger("Оберіть код операції: ", action);
		system("cls");
		switch (action)
		{
		case 1:
		{
			printData(competitionList);
			pause();
			break;
		}
		case 2:
		{
			printData(competitionList);
			do
			{
				cout << "Введіть значення від 1 до " <<
competitionList.elementCount + 1 << endl;
				getInteger("Введіть позицію елемента для додавання (0 для
скасування): ", number);
			} while (number < 0 || number > competitionList.elementCount + 1);
			system("cls");
			if (number != 0)
				addNode(competitionList, getNode(), number);
			else
				cout << "Операцію скасовано" << endl;
			pause();
			break;
		}
		case 3:
		{
			printData(competitionList);
			if (competitionList.elementCount != 0)
			{
				do
				{
					cout << "Ви можете видалити елемент від 1 до " <<
competitionList.elementCount << endl;
					getInteger("Введіть позицію елемента для видалення (0
для скасування): ", number);
				} while (number < 0 || number > competitionList.elementCount);
```

```
                        if (number != 0)
                                deleteNode(competitionList, number);
                        else
                                cout << "Операцію скасовано" << endl;
                }
                pause();
                break;
        }
        case 4:
        {
                printData(competitionList);
                do
                {
                        cout << "Ви можете змінити елемент від 1 до " <<
competitionList.elementCount << endl;
                        getInteger("Введіть позицію елемента для коригування (0 для
скасування): ", number);
                } while (number < 0 || number > competitionList.elementCount);
                if (number != 0)
                        editNode(competitionList, number);
                else
                        cout << "Операцію скасовано" << endl;
                pause();
                break;
        }
        case 5:
        {
                sportsmenData(competitionList);
                pause();
                break;
        }
        case 6:
        {
                getString("Введіть назву спортивної дисципліни: ", tmp);
                sportSearch(competitionList, tmp);
                pause();
                break;
        }
        case 7:
        {
                newBest(competitionList);
                pause();
                break;
        }
        case 8:
        {
                getString("Введіть назву країни: ", tmp);
                countrySearch(competitionList, tmp);
                pause();
                break;
        }
        case 9:
```

```
		{
			getCountryResults(competitionList);
			pause();
			break;
		}
		case 10:
		{
			sortByName(competitionList);
			pause();
			break;
		}
		case 11: saveToFile(competitionList); pause();  break;
		case 12: break;
		default: break;
		}
	} while (action != 11 && action != 12);
	return 0;
}

//Ожидание нажатия любой клавиши
void pause()
{
	cout << endl << "Натисніть будь-яку клавішу, щоб продовжити...";
	cin.ignore();
	system("cls");
}

//Ручной ввод данных
ListNode* getNode()
{
	ListNode* node = new ListNode;
	printSeparator('-');
	getString("Введіть назву виду спорту:                              | ", node->sportName);
	printSeparator('-');
	getString("Введіть ім'я спортсмена, що посів перше місце:          | ", node-
>firstPlaceName);
	printSeparator('-');
	getString("Введіть країну спортсмена, що посів перше місце:        | ", node-
>firstPlaceCountry);
	printSeparator('-');
	getDouble("Введіть результат спортсмена, що посів перше місце:     | ", node-
>firstPlaceResult);
	printSeparator('-');
	getDouble("Введіть персональний рекорд спортсмена, що посів перше місце:  | ",
node->firstPlaceBest);
	printSeparator('-');
	getString("Введіть ім'я спортсмена, що посів друга місце:          | ", node-
>secondPlaceName);
	printSeparator('-');
	getString("Введіть країну спортсмена, що посів друга місце:        | ", node-
>secondPlaceCountry);
	printSeparator('-');
```

```
	getDouble("Введіть результат спортсмена, що посів друга місце:		| ", node->secondPlaceResult);
	printSeparator('-');
	getDouble("Введіть персональний рекорд спортсмена, що посів друга місце:	| ", node->secondPlaceBest);
	printSeparator('-');
	getString("Введіть ім'я спортсмена, що посів третє місце:		| ", node->thirdPlaceName);
	printSeparator('-');
	getString("Введіть країну спортсмена, що посів третє місце:		| ", node->thirdPlaceCountry);
	printSeparator('-');
	getDouble("Введіть результат спортсмена, що посів третє місце:		| ", node->thirdPlaceResult);
	printSeparator('-');
	getDouble("Введіть персональний рекорд спортсмена, що посів третє місце:	| ", node->thirdPlaceBest);
	printSeparator('-');
	return node;
}


void printData(ListHead& listHead)
{

	if (listHead.first == nullptr)
	{
		printf("Список пустий!");
	}
	else
	{
		printSeparator('-');
		printf("  Назва спортивної дисципліни  | №  |	Ім'я спортсмена	|");
		printf("	Країна	|	Результат	| Особистий рекорд |\n");
		ListNode* current = listHead.first;
		printSeparator('-');
		while (current != nullptr)
		{
			printNode(current);
			current = current->next;
		}
	}
}

//Вывести на экран данные одного узла списка
void printNode(ListNode* node)
{
	printf(" %-30s | № 1 | %-30s | %-30s | Результат: %10.2lf | Рекорд: %10.2lf |\n",
		node->sportName,
		node->firstPlaceName,
		node->firstPlaceCountry,
		node->firstPlaceResult,
```

```
                node->firstPlaceBest);
        printf(" %-30s | № 2 | %-30s | %-30s | Результат: %10.2lf | Рекорд: %10.2lf |\n",
                "",
                node->secondPlaceName,
                node->secondPlaceCountry,
                node->secondPlaceResult,
                node->secondPlaceBest);
        printf(" %-30s | № 3 | %-30s | %-30s | Результат: %10.2lf | Рекорд: %10.2lf |\n",
                "",
                node->thirdPlaceName,
                node->thirdPlaceCountry,
                node->thirdPlaceResult,
                node->thirdPlaceBest);
        printSeparator('-');

}

//Функция безопасного ввода дробного числа
void getDouble(const char* message, double& number)
{
        cout << message;
        cin >> number;
        while (cin.fail())
        {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Помилка введення." << endl;
                cout << message;
                cin >> number;
        }
        cin.ignore();
}

//Функция безопасного ввода целого числа
void getInteger(const char* message, int& number)
{
        cout << message;
        cin >> number;
        while (cin.fail())
        {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "Помилка введення." << endl;
                cout << message;
                cin >> number;
        }
        cin.ignore();
}

//Функция безопасного ввода строки
void getString(const char* message, char* stringValue)
{
```

```
	cout << message;
	cin.getline(stringValue, STR_LENGTH);
	if (cin.fail())
	{
		cin.clear();
		cin.ignore(numeric_limits<streamsize>::max(), '\n');
	}
}

//Добавить элемент в список
void addNode(ListHead& listHead, ListNode* node, int position)
{
	listHead.elementCount++;
	node->next = nullptr;
	if (listHead.first == nullptr)
	{
		listHead.first = node;
	}
	else
	{
		ListNode* current = listHead.first;
		ListNode* previous = nullptr;
		for (int i = 1; i < position; i++)
		{
			previous = current;
			current = current->next;
		}
		node->next = current;
		(previous == nullptr) ? (listHead.first = node) : (previous->next = node);
	}
}

//Вывести главное меню
void printMenu()
{
	system("cls");
	printf("%62sСписок доступних варіантів\n", "");
	printSeparator('~');
	printf("%20s 1) Вивести список на екран	2) Додати новий запис	3) 
Видалити існуючий запис\n", "");
	printf("%20s 4) Коригувати дані		5) Дані про всіх спортсменів	6) Дані за 
видом спорту\n", "");
	printf("%20s 7) Спортсмени з новим рекордом	8) Пошук за країною	9) 
Рейтинг країн\n", "");
	printf("%20s10) Сортування за назвою спорту	11) Зберегти та вийти	12) 
Вийти без збереження\n", "");
	printSeparator('~');
}

//Сохранить список в бинарный файл
void saveToFile(ListHead& listHead)
{
```

```
	FILE* file = nullptr;
	ListNode* current = listHead.first;
	if ((file = fopen("data.bin", "wb")) == nullptr)
	{
		cout << "Помилка відкриття файлу, збереження неможливе!" << endl << endl;
	}
	else
	{
		while (current != nullptr)
		{
			fwrite(current, sizeof(ListNode) - sizeof(ListNode*), 1, file);
			current = current->next;
		}
		fclose(file);
	}
}

//Считать данные с бинарного файла
void readFromFile(ListHead& listHead)
{
	FILE* file;
	if ((file = fopen("data.bin", "rb")) == nullptr)
	{
		cout << "Помилка читання файлу!" << endl << endl;
	}
	else
	{

		ListNode* previous = nullptr;
		while (!feof(file))
		{
			ListNode* current = new ListNode;
			fread(current, sizeof(ListNode) - sizeof(ListNode*), 1, file);
			if (!feof(file))
			{
				addNode(listHead, current, listHead.elementCount + 1);
			}
			else delete current;
		}
	}
}

//Удалить элемент списка по индексу
void deleteNode(ListHead& listHead, int number)
{
	ListNode* current = listHead.first;
	ListNode* previous = nullptr;
	listHead.elementCount--;
	for (int i = 1; i < number; i++)
	{
		previous = current;
		current = current->next;
```

```
        }
        (previous == nullptr) ? (listHead.first = current->next) : (previous->next = current->next);
        delete current;
}

//Получение информации о всех спортсменах
void sportsmenData(ListHead& listHead)
{
        char** sportsmen = new char* [listHead.elementCount];
        for (int i = 0; i < listHead.elementCount; i++)
        {
                sportsmen[i] = new char[STR_LENGTH];
        }
        int k = 0;
        ListNode* current = listHead.first;
        while (current != nullptr)
        {
                bool isHere = false;
                for (int i = 0; i < k; i++)
                {
                        if (strcmp(current->firstPlaceName, sportsmen[i]) == 0)
                        {
                                isHere = true;
                                break;
                        }
                }
                if (!isHere) sportsmen[k++] = current->firstPlaceName;
                isHere = false;
                for (int i = 0; i < k; i++)
                {
                        if (strcmp(current->secondPlaceName, sportsmen[i]) == 0)
                        {
                                isHere = true;
                                break;
                        }
                }
                if (!isHere) sportsmen[k++] = current->secondPlaceName;
                isHere = false;
                for (int i = 0; i < k; i++)
                {
                        if (strcmp(current->thirdPlaceName, sportsmen[i]) == 0)
                        {
                                isHere = true;
                                break;
                        }
                }
                if (!isHere) sportsmen[k++] = current->thirdPlaceName;
                current = current->next;
        }
        for (int i = 0; i < k; i++)
        {
                printSeparator('-');
```

```
                printf(" %-30s", sportsmen[i]);
                bool isFirst = true;
                current = listHead.first;
                while (current != nullptr)
                {
                        if (strcmp(current->firstPlaceName, sportsmen[i]) == 0) {
                                if (!isFirst) printf(" %30s", "");
                                printf("  |%-30s |", current->sportName);
                                printf(" 1 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n",
current->firstPlaceResult, current->firstPlaceBest);
                                isFirst = false;
                        }
                        if (strcmp(current->secondPlaceName, sportsmen[i]) == 0) {
                                if (!isFirst) printf(" %30s", "");
                                printf("  |%-30s |", current->sportName);
                                printf(" 2 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n",
current->secondPlaceResult, current->secondPlaceBest);
                                isFirst = false;
                        }
                        if (strcmp(current->thirdPlaceName, sportsmen[i]) == 0) {
                                if (!isFirst) printf(" %30s", "");
                                printf("  |%-30s |", current->sportName);
                                printf(" 3 місце | Результат: %10.2lf | Рекорд: %10.2lf |\n",
current->thirdPlaceResult, current->thirdPlaceBest);
                                isFirst = false;
                        }
                        current = current->next;
                }

        }
        printSeparator('-');
}

//Сортировать сортивные дисциплины по названию
void sortByName(ListHead& listHead)
{
        ListNode* previous;
        ListNode* current;
        ListNode* next;
        for (int i = 1; i <= listHead.elementCount; i++)
        {
                current = listHead.first;
                previous = nullptr;
                for (int j = 1; j <= listHead.elementCount - i; j++)
                {
                        if (current != nullptr)
                        {
                                if (current->next != nullptr)
                                {
                                        next = current->next;
                                        if (next != nullptr && strcmp(current->sportName, next-
>sportName) > 0)
```

```
                                        {
                                                if (previous != nullptr) previous->next = next;
                                                else listHead.first = next;
                                                current->next = next->next;
                                                next->next = current;
                                                current = next;
                                        }
                                        previous = current;
                                        current = current->next;
                                }
                        }
                }
        }

}

//Напечатать разделитель
void printSeparator(const char symb)
{
        for (int i = 0; i < 150; i++) printf("%c", symb);
        printf("\n");
}

//Редактировать данные элемента по индексу
void editNode(ListHead& listHead, int number)
{
        ListNode* current = listHead.first;
        for (int i = 1; i < number; i++)
        {
                current = current->next;
        }
        printSeparator('-');
        printf("%65s >>>>>>> ", current->sportName);
        getString("", current->sportName);
        printSeparator('-');
        printf("%75s\n", "1 місце");
        printSeparator('-');
        printf("%70s ", current->firstPlaceName);
        getString(">>>>>>> ", current->firstPlaceName);
        printSeparator('-');
        printf("%70s ", current->firstPlaceCountry);
        getString(">>>>>>> ", current->firstPlaceCountry);
        printSeparator('-');
        printf("%70.2lf ", current->firstPlaceResult);
        getDouble(">>>>>>> ", current->firstPlaceResult);
        printSeparator('-');
        printf("%70.2lf ", current->firstPlaceBest);
        getDouble(">>>>>>> ", current->firstPlaceBest);

        printSeparator('-');
        printf("%75s\n", "2 місце");
        printSeparator('-');
```

```
	printf("%70s ", current->secondPlaceName);
	getString(">>>>>>> ", current->secondPlaceName);
	printSeparator('-');
	printf("%70s ", current->secondPlaceCountry);
	getString(">>>>>>> ", current->secondPlaceCountry);
	printSeparator('-');
	printf("%70.2lf ", current->secondPlaceResult);
	getDouble(">>>>>>> ", current->secondPlaceResult);
	printSeparator('-');
	printf("%70.2lf ", current->secondPlaceResult);
	getDouble(">>>>>>> ", current->secondPlaceResult);

	printSeparator('-');
	printf("%75s\n", "3 місце");
	printSeparator('-');
	printf("%70s ", current->thirdPlaceName);
	getString(">>>>>>> ", current->thirdPlaceName);
	printSeparator('-');
	printf("%70s ", current->thirdPlaceCountry);
	getString(">>>>>>> ", current->thirdPlaceCountry);
	printSeparator('-');
	printf("%70.2lf ", current->thirdPlaceResult);
	getDouble(">>>>>>> ", current->thirdPlaceResult);
	printSeparator('-');
	printf("%70.2lf ", current->thirdPlaceResult);
	getDouble(">>>>>>> ", current->thirdPlaceResult);
	printSeparator('-');

}

//Поиск спортивной дисциплины по названию
void sportSearch(ListHead& listHead, char* sportName)
{
	ListNode* current = listHead.first;
	int isPrinted = false;
	while (current != nullptr) {
		if (strcmp(current->sportName, sportName) == 0)
		{
			if (!isPrinted) printSeparator('-');
			printNode(current);
			isPrinted = true;
		}
		current = current->next;
	}
	if (!isPrinted)
		cout << "Нічого не знайдено!" << endl << endl;
}

//Поиск спотсменов по стране
void countrySearch(ListHead& listHead, char* countryName)
{
	char** sportsmen = new char* [listHead.elementCount];
```

```
for (int i = 0; i < listHead.elementCount; i++)
{
	sportsmen[i] = new char[STR_LENGTH];
}
int k = 0;
ListNode* current = listHead.first;
while (current != nullptr)
{
	bool isHere = true;

	if (strcmp(current->firstPlaceCountry, countryName) == 0)
	{
		isHere = false;
		for (int i = 0; i < k; i++)
		{
			if (strcmp(current->firstPlaceName, sportsmen[i]) == 0)
			{
				isHere = true;
				break;
			}
		}
	}
	if (!isHere) sportsmen[k++] = current->firstPlaceName;
	isHere = true;
	if (strcmp(current->secondPlaceCountry, countryName) == 0)
	{
		isHere = false;
		for (int i = 0; i < k; i++)
		{
			if (strcmp(current->secondPlaceName, sportsmen[i]) == 0)
			{
				isHere = true;
				break;
			}
		}
	}
	if (!isHere) sportsmen[k++] = current->secondPlaceName;
	isHere = true;
	if (strcmp(current->thirdPlaceCountry, countryName) == 0)
	{
		isHere = false;
		for (int i = 0; i < k; i++)
		{
			if (strcmp(current->thirdPlaceName, sportsmen[i]) == 0)
			{
				isHere = true;
				break;
			}
		}
	}
	if (!isHere) sportsmen[k++] = current->thirdPlaceName;
	current = current->next;
```

```
	}
	if (k == 0)
	{
		cout << "Спортсменів в заданій країні не знайдено!" << endl << endl;
	}
	else
	{
		for (int i = 0; i < k; i++)
		{
			printSeparator('-');
			cout << sportsmen[i] << endl;
		}
		printSeparator('-');
	}

}

//Поиск спортсменов, улучшевших свой результат
void newBest(ListHead& listHead)
{
	ListNode* current = listHead.first;
	bool isPrinted = false;
	while (current != nullptr)
	{
		if (current->firstPlaceResult < current->firstPlaceBest)
		{
			printSeparator('-');
			printf(" %-30s | %-30s | %-30s |   %-10.2lf -> %10.2lf\n",
				current->sportName,
				current->firstPlaceName,
				current->firstPlaceCountry,
				current->firstPlaceBest,
				current->firstPlaceResult);
			isPrinted = true;
		}
		if (current->secondPlaceResult < current->secondPlaceBest)
		{
			printSeparator('-');
			printf(" %-30s | %-30s | %-30s |   %-10.2lf -> %10.2lf\n",
				current->sportName,
				current->secondPlaceName,
				current->secondPlaceCountry,
				current->secondPlaceBest,
				current->secondPlaceResult);
			isPrinted = true;
		}
		if (current->thirdPlaceResult < current->thirdPlaceBest)
		{
			printSeparator('-');
			printf(" %-30s | %-30s | %-30s |   %-10.2lf -> %10.2lf\n",
				current->sportName,
				current->thirdPlaceName,
```

```
                        current->thirdPlaceCountry,
                        current->thirdPlaceBest,
                        current->thirdPlaceResult);
                isPrinted = true;
            }
            current = current->next;
        }

        if (!isPrinted)
            cout << "Нових рекордів не зафіксовано!" << endl;
        else printSeparator('-');
}

//Получить информацию об общих результатах страны
void getCountryResults(ListHead& listHead)
{
        struct CountryResults
        {
            char country[STR_LENGTH];
            int firstCount = 0;
            int secondCount = 0;
            int thirdCount = 0;
        } *countryResult = new CountryResults[listHead.elementCount * 3];

        ListNode* current = listHead.first;
        int k = 0;
        while (current != nullptr)
        {
            bool isHere = false;
            for (int i = 0; i < k; i++)
            {
                if (strcmp(current->firstPlaceCountry, countryResult[i].country) == 0)
                {
                    countryResult[i].firstCount++;
                    isHere = true;
                    break;
                }
            }
            if (!isHere)
            {
                strcpy(countryResult[k].country, current->firstPlaceCountry);
                countryResult[k].firstCount = 1;
                k++;
            }
            isHere = false;
            for (int i = 0; i < k; i++)
            {

                if (strcmp(current->secondPlaceCountry, countryResult[i].country) == 0)
                {
                    countryResult[i].secondCount++;
                    isHere = true;
```

```
                        break;
                    }
                }
                if (!isHere)
                {
                    strcpy(countryResult[k].country, current->secondPlaceCountry);
                    countryResult[k].secondCount = 1;
                    k++;
                }
                isHere = false;
                for (int i = 0; i < k; i++)
                {
                    if (strcmp(current->thirdPlaceCountry, countryResult[i].country) == 0)
                    {
                        countryResult[i].thirdCount++;
                        isHere = true;
                        break;
                    }
                }
                if (!isHere)
                {
                    strcpy(countryResult[k].country, current->thirdPlaceCountry);
                    countryResult[k].thirdCount = 1;
                    k++;
                }
                current = current->next;
            }
            if (k == 0) cout << "Нічого не знайдено!" << endl << endl;
            else printSeparator('-');
            for (int i = 0; i < k; i++)
            {
                printf(" %-30s | Золото: %3d | Срібло: %3d | Бронза: %3d |\n",
                    countryResult[i].country,
                    countryResult[i].firstCount,
                    countryResult[i].secondCount,
                    countryResult[i].thirdCount);
                printSeparator('-');
            }


}
```